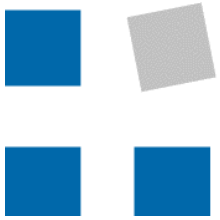


HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

XML

Teil 1: Grundbegriffe

Abteilung Informatik – WS 02/03

Was ist XML?

- XML-Anwendung: eine XML-Auszeichnungssprache

Was ist XML nicht?

XML ist keine...

- ... Programmiersprache
- ... Netzwerk-Kommunikationsprotokoll
- ... Datenbank

Beispiel

```
<?xml version="1.0" ?>
<product barcode="2394287410">
  <manufacturer>Verbatim</manufacturer>
  <name>DataLife MF 2HD</name>
  <quantity>10</quantity>
  <size>3.5"</size>
  <color>black</color>
  <description>floppy disks</description>
</product>
```

XML-Dokument

- Ein XML-Dokument ist ein Text (keine binäre Daten) und könnte als Text-File gespeichert werden
- Ein XML-Dokument kann mit einem üblichen Text-Editor (z.B. TextPad, vi) editiert werden
- Ein XML-Dokument kann:
  - Ein File sein
  - Ein Record oder ein Feld in einer Datenbank sein
  - Von einem CGI-Programm „on the fly“ erzeugt werden
  - In mehrere Files gespeichert werden
  - ...

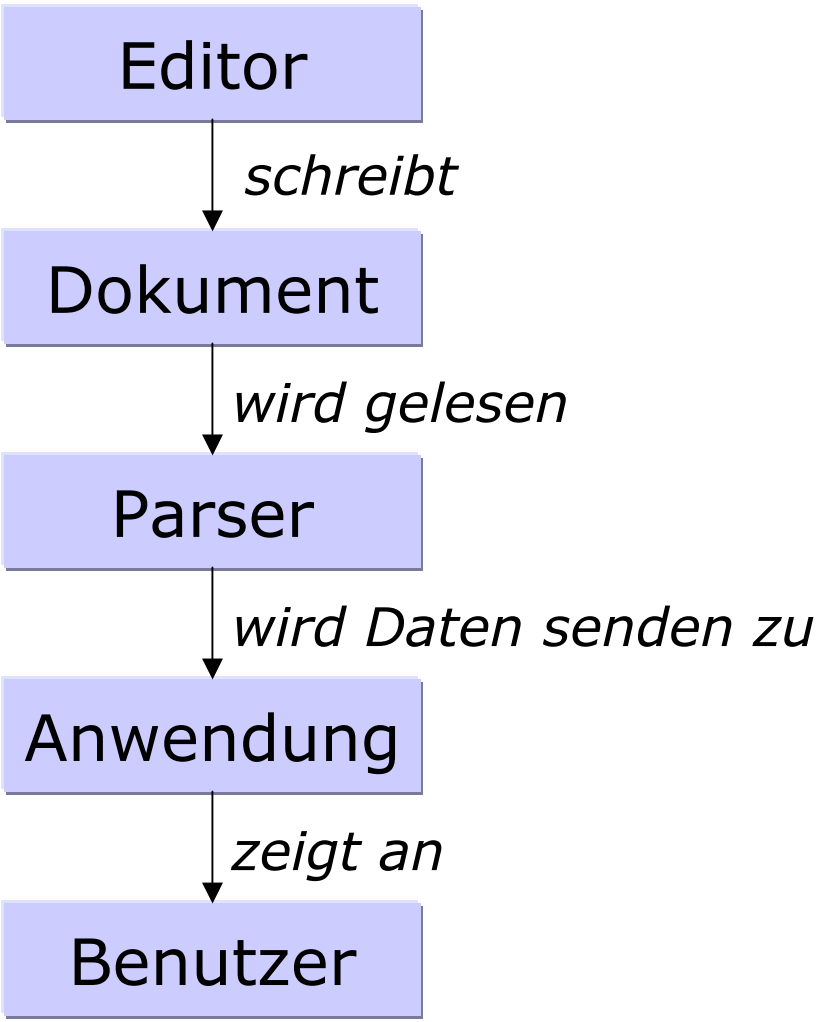
XML-Parser

- XML-Parser (= XML-Prozessor) liest das Dokument und überprüft
  - ob der XML-Code wohlgeformt ist, d.h. ob er den XML-Spezifikationen erfüllt
  - ob der Code gültig ist (optional), d.h. ob er die Anforderungen des entsprechenden Schemas (z.B. DTD) erfüllt
- Der Parser zerlegt das Dokument in Elemente, Attribute, etc. und übergibt den Inhalt des Dokuments

XML-Parser: Fehlermeldungen

- Wohlgeformtheitsfehler: Das Dokument ist nicht wohlgeformt
  - Fehlermeldung
  - Abbruch
- Gültigkeitsfehler: Das Dokument ist nicht gültig
  - Fehlermeldung an die entsprechende Anwendung (z.B. Browser)
  - Die Anwendung entscheidet über den Abbruch oder die Fortsetzung des Parsing-Prozesses

Lebenszyklus eines XML-Dokument



Element

- Ein (nicht leeres) **Element** beginnt mit einem **Start-Tag** (= öffnender Tag) und endet mit einem **End-Tag** (schliessender Tag)
- Der **Inhalt** des Elements wird durch den Start- und End-Tag begrenzt

Beispiel: `<person>Alan Turing</person>`

Achtung: XML unterscheidet Gross- und Kleinschreibung (engl. „case sensitive“)

Tag-Syntax

- Ein Start-Tag begint mit < und endet mit >
- Ein End-Tag beginnt mit </ und endet mit >
- Leeres Element:
  - Normaler Start- und End-Tag, leerer Inhalt
  - Ein einziger Tag, welcher mit < beginnt und /> endet

Tag-Syntax: Beispiele

- Start-Tag: `<person>`
- End-Tag: `</person>`
- Leeres Element-Tag: `<br/>` (äquivalent zu `<br></br>`)

Elementen-Verschachtelung: Beispiel

```
<person>
 <name>
  <first_name>Alan</first_name>
  <last_name>Turing</last_name>
 </name>
  <profession>computer scientist</profession>
  <profession>mathematician</profession>
  <profession>cryptographer</profession>
</person>
```

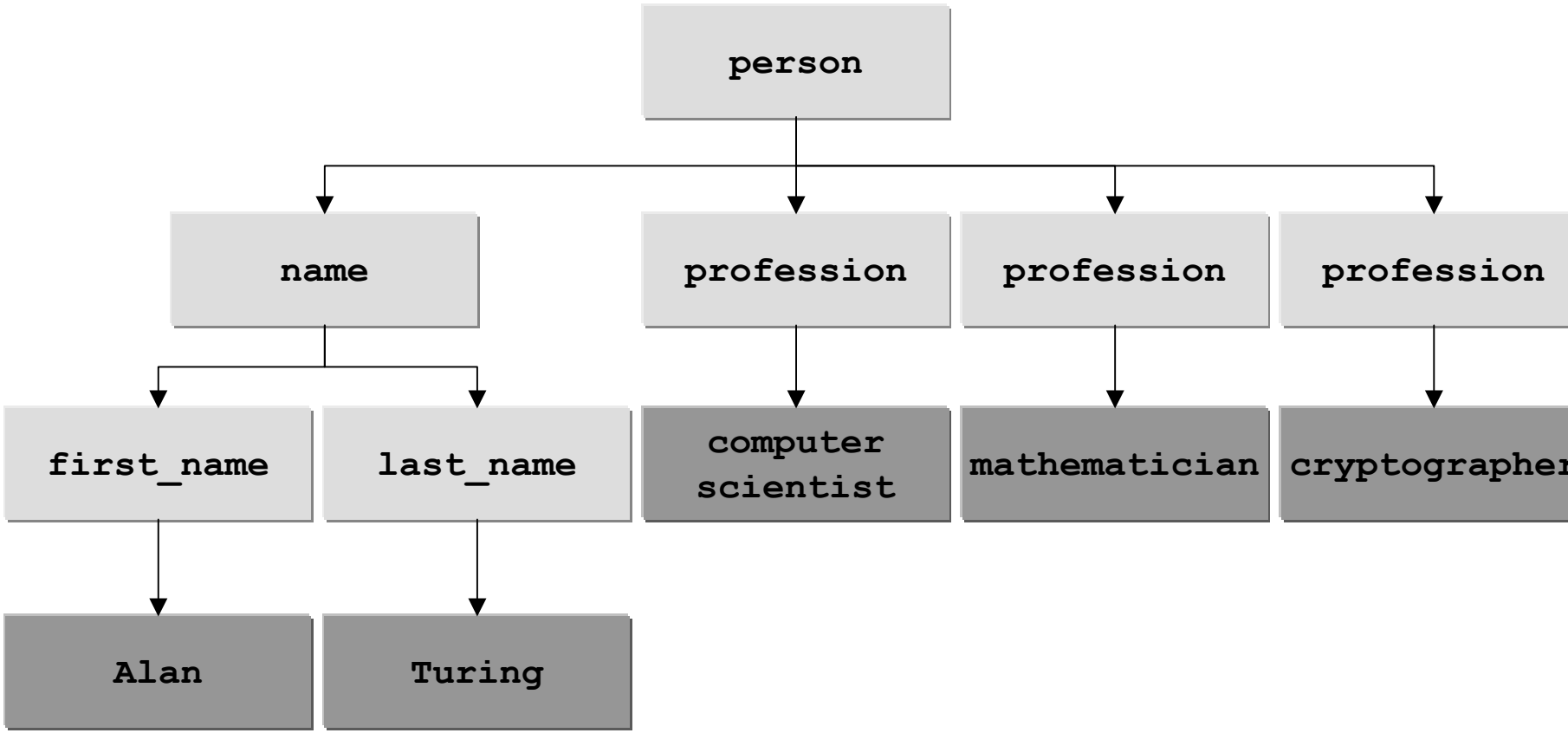
Elementen-Verschachtelung

- Der Inhalt eines Elements besteht aus weiteren Elementen, aus Text, oder aus beiden
- Elementen-Hierarchie:

```
<name>
        <first_name>Alan</first_name>
        <last_name>Turing</last_name>
</name>
```

  - `name`: „Parent"-Element von `first_name` und `last name`
  - `first_name`: „Child"-Element von `name`
  - `first name` und `last_name` sind „Sibling"-Elemente (Sibling = Geschwister)
- Überlappung von Tags, wie z.B. `<name><first_name>Alan </name>< /first_name>`) ist verboten

XML-Baum: Beispiel



Root-Element

- Ein Element ohne „Parent"-Element heisst **„Root"-Element**
- Jeder XML-Dokument hat **genau ein** Root-Element (= Dokument-Element)

Attribut

- **Attribut** = Name-Wert-Paar, das mit einem Element verbunden ist
- Attribute sind im Start-Tag angegeben
- Attributsyntax:

```
Attributname = "Attributwert"
```

oder:

```
Attributname = 'Attributwert'
```

- Beispiel:

```
<person born="1912-06-23" died="1954-06-07">
    Alan Turing
</person>
```

Attribute oder Child-Elemente? Beispiel

- Mit Attributen:

```
<person>
 <name first_name="Alan" last_name="Turing"></name>
 <profession value="computer scientist"/>
 <profession value="mathematician"/>
 <profession value="cryptographer"/>
</person>
```

- Mit Child-Elementen:

```
<person>
 <name>
  <first_name>Alan</first_name>
  <last_name>Turing</last_name>
 </name>
  <profession>computer scientist</profession>
  <profession>mathematician</profession>
  <profession>cryptographer</profession>
</person>
```

Attribute oder Child-Elementen?

- Es gibt keine grundsätzliche Regel, die bestimmen würde, wann Child-Elemente und wann Attribute verwendet werden sollten
- Faustregel:
  - Daten sollen in Elemente gespeichert werden.
  - Informationen über Daten (=Metadaten, z.B. ID, URL, Referenz) sollen in Attributen gespeichert werden
  - *Im Zweifelsfall sollen Informationen in Elementen speichern*
- Es gibt Fälle, wo Meta-Daten in Elemente gespeichert werden sollen:
  - Strukturierte Meta-Daten
  - Speichern von Meta-Metadaten
  - Bessere Flexibilität (z.B. im Hinblick auf zukünftige Änderungen)

XML-Namen

- XML-Name: Elementname, Attributname, ...
- XML-Name dürfen folgende Zeichen enthalten:
  - Alphanumerische Zeichen: A B C .. Y Z a b c .. y z 0 1 ..9
  - Weitere Buchstaben: ç, ö, Φ, ...
  - _ (Unterstrich)
  - - (Bindestrich)
  - . (Punkt)
  - : (Doppelpunkt, jedoch mit Einschränkungen, s. Namensräume)
- XML-Namen fangen mit einem Buchstaben, einem Ideogramm oder einem Unterstrich an
- Weitere Satzzeichen wie [, ; & % etc. sind verboten
- Es gibt keine obere Grenze für die Länge eines XML-Namens

Entitätsreferenzen

- Entität = Speichereinheit, die einen bestimmten Teil eines XML-Dokuments enthält (z.B. Datei, Datenbank-Datensatz, etc.)
- Es gibt fünf vordefinierte Entitätsreferenzen. Sie beziehen sich auf Zeichen (Textentität)

`&` für `&`

`<` für `<`

`>` für `>` (optional, d.h. > darf auch verwendet werden)

`&quot;` für `"` (optional, d.h. " darf auch verwendet werden)

`&apos;` für `'` (optional, d.h. ' darf auch verwendet werden)

Entitätsreferenzen: Beispiele

```
<CODE> for (int i = 0; i &lt;= args.length; i++) </CODE>

<CODE> for (int i = args.length; i &gt;=0; i--) </CODE>

<CODE> for (int i = args.length; i >= 0; i--) </CODE>
```

CDATA-Abschnitt „character data“

- Ein CDATA-Abschnitt beginnt mit `<![CDATA[` und endet mit `]]>`
- Was zwischen diesen zwei Aufzeichnungen steht, wird als *rohe* Daten verarbeitet
- Beispiel:

```
<![CDATA[
     for (int i = 0; i <= args.length; i++)
]]>
```

- `]]>` ist die einzige Zeichenfolge, die in einem CDATA-Abschnitt nicht vorkommen darf
- Daten im CDATA Abschnitt werden nicht geparsed

Kommentare

- Ein XML-Kommentar beginnt mit <!–- und endet mit -->
- Ein Kommentar **darf** vor dem Root-Element plaziert werden
- Ein Kommentar **darf nicht** in einem Tag vorkommen

XML-Deklaration

- Ein XML-Dokument kann (muss aber nicht) mit einer XML-Deklaration anfangen
- Wenn vorhanden, ist die XML-Deklaration immer die erste Zeile des XML-Dokuments
  - Zum Beispiel dürfen keine Kommentare, keine Anweisungen, kein Leerzeichen etc. vor der XML-Deklaration vorkommen (eventuell darf eine nicht sichtbare Byte-Ordermarke vorkommen)
- Eine XML-Deklaration hat drei Attribute: `version`, `standalone`, und `encoding`
- Beispiel:

```
<?xml version= "1.0" encoding= "ASCII" standalone= "yes"?>
```

Attribute der XML-Deklaration

- `version`: z.Z. nur `1.0`
- `encoded`: (Wert: `ISO-8859-1`, `ISO-2022-JP`, ...) Zeichenkodierung des Dokuments
- `standalone` (Wert: `yes|no`): gibt es Referenzen zu externen Entitäten?

Zeichensatz

- Ein **Zeichensatz** ordnet Zahlen Charaktere (z.B. Buchstaben) zu
- Beispiele von Zeichensätzen: ASCII, Unicode
- Beispiel: Unicode

  $A \rightarrow 65$ $B \rightarrow 66$ $\Sigma \rightarrow 931$
- Eine zu einem Charakter zugeordnete Zahl heisst der **Kodierungspunkt** (engl. Code point) dieses Charakters
- Beispiel: `65` ist der Kodierungspunkt von `A`

Unicode

- ISO-Standard
- Version 3.2 enthält 95 156 Charaktere
- ASCII ist eine Untermenge von Unicode

Zeichenkodierung

- Eine **Zeichenkodierung** (engl. Character encoding) beschreibt, wie die Kodierungspunkte in Bytes dargestellt werden
- Einige Zeichensätze (z.B. ASCII) habe nur eine Kodierung
- Beispiele von Zeichenkodierung von Unicode: UCS-2, UCS-4, UTF-8, UTF-16

UCS-2

- Vollständige Name: ISO-10646-UCS-2
- UCS-2 ist eine Zeichensatzkodierung von Unicode. Jeder Charakter ist durch eine ganze Zahl ohne Vorzeichen zwischen 0 und 65 635 (d.h. 2 Bytes) dargestellt. Beispiel: Σ (Kodierungspunkt 931) ist durch die Bytes 03 und A3 dargestellt (Hex).
- Zwei Varianten
  - „Little endian": Σ → #xA303 („less significant" Byte zuerst)
  - „Big endian": Σ → #x03A3 („most significant" Byte zuerst)
- Die Kodierungsvariante ist durch die **Byte-Ordermarke** (engl. byte-order mark) angegeben, welche dem Kodierungspunkt #xFEFF („zero width non-breaking space") entspricht

XML-Anwendung

- XML: Meta-Auszeichnungsprache
- XML-Anwendung (engl. XML-Application): XML-basierte Auszeichnungssprache
- Achtung: XML-Anwendung (z.B. MathML) ≠ Anwendung welche XML verwendet (z.B. Browser)

Referenzen

- E. Harold, W. Means: *XML in a Nutshell*, Kapitel 1, 2